Slicing and Dicing Stories for the Product Backlog

Alan O'Callaghan

Director and Principal Consultant

Emerald Hill Limited

Suite 4

The Lawns Business Centre

The Lawns

Hinckley, Leicester

Leicestershire LE10 1DY

ajocallaghan@gmail.com

1. Abstract

This paper presents a group of patterns intended for a future pattern language describing better ways of using stories as Product Backlog items in Scrum. While it recognizes and embraces prior work in this area the paper focuses on the particular problems faced by Product Owners and Scrum Teams when stories are used as Product Backlog Items. The author recently has had repeated experience of Teams burning in 'story card hell' and it is these experiences which have prompted this effort to clarify further the thinking, and practice, of the use of stories.

2. Introduction

Stories are a widely-used technique in Agile software development. Originating with eXtreme Programming (XP), they have spread to some degree into every Agile method and framework. There is a substantial body of literature surrounding the practical use of stories. Mike Cohn's User Stories Applied (1) is probably the best known work. It includes references to Ron Jeffries' Three C's (2) -Card, Conversation, Confirmation- which reminds us to write the stories as brief sentences on index cards; to use them to drive conversations between developers; and to confirm the acceptance criteria by which the users and customers can judge their successful implementation. Cohn also describes the well-known acronym, INVEST, by which Bill Wake characterizes "good" stories: Independent, Negotiable, Valuable, Estimatable, Small and Testable (3).

The first version of The Scrum Guide noted that "Product Backlog items are usually stated as User Stories" (4). More than 70% of Agile teams claim to be using Scrum (5). Although there are no references to stories in the subsequent versions (6), that statement is probably more true than ever. But that very popularity in Scrum has raised new issues. The Product Owner in Scrum is charged with ordering the Product Backlog items (PBIs) top to bottom and it is clear that some Scrum teams have been descending into "story card hell" whereby the Product Backlog is so crammed with fine-grained stories of all types that it is becoming extremely difficult to rationalise decision-making about which stories to prioritize. The author has experience of Scrum teams complaining that they “don’t know where to start” when faced with this situation. It is the apparently increasing occurrence of “story card hells” that has prompted the patterns described below.

Richard Lawrence published nine patterns for splitting stories in 2009 (9). These were Workflow Steps, Business Rule Variations, Major Effort, Simple/Complex, Variations in Data, Date Entry Methods, Defer Performance, Operations (e.g., CRUD) and Break Out a Spike. Dean Leffingwell added a tenth, called Use-Case Scenarios to this group in his book, Agile Software Requirements (10). The four patterns described below are not designed to replace Lawrence’s (although Minimal Marketable Feature shares some common ground with Lawrence’s Simple/Complex pattern) but to complement them and to draw together, with them, a sequence of patterns that can guide Scrum practitioners, Product Owners in particular, and Agile teams in general to better management of their various Backlogs when using stories as PBIs.

The intention is to contribute to the creation of a true pattern language of Story patterns revisiting, if necessary, previously published patterns to create a more cohesive and accessible set. The current thinking behind this project is reflected in the Mind Map in Figure 1. The patterns included in this paper for workshopping are coloured blue in the Mind Map and are presented in a format based on the Portland format (11). This format has been extended in two ways: first, a ‘Known Uses’ clause has been added, and, second, the ‘Problem” clause - formulated as a question - and the first sentence or two of the ‘Solution’ clause have been emboldened. The intention here is that these emboldened sentences can, in the future, form the basis of thumbnails for each pattern in the language providing a searchable, shorthand description.

3. Story Patterns MindMap

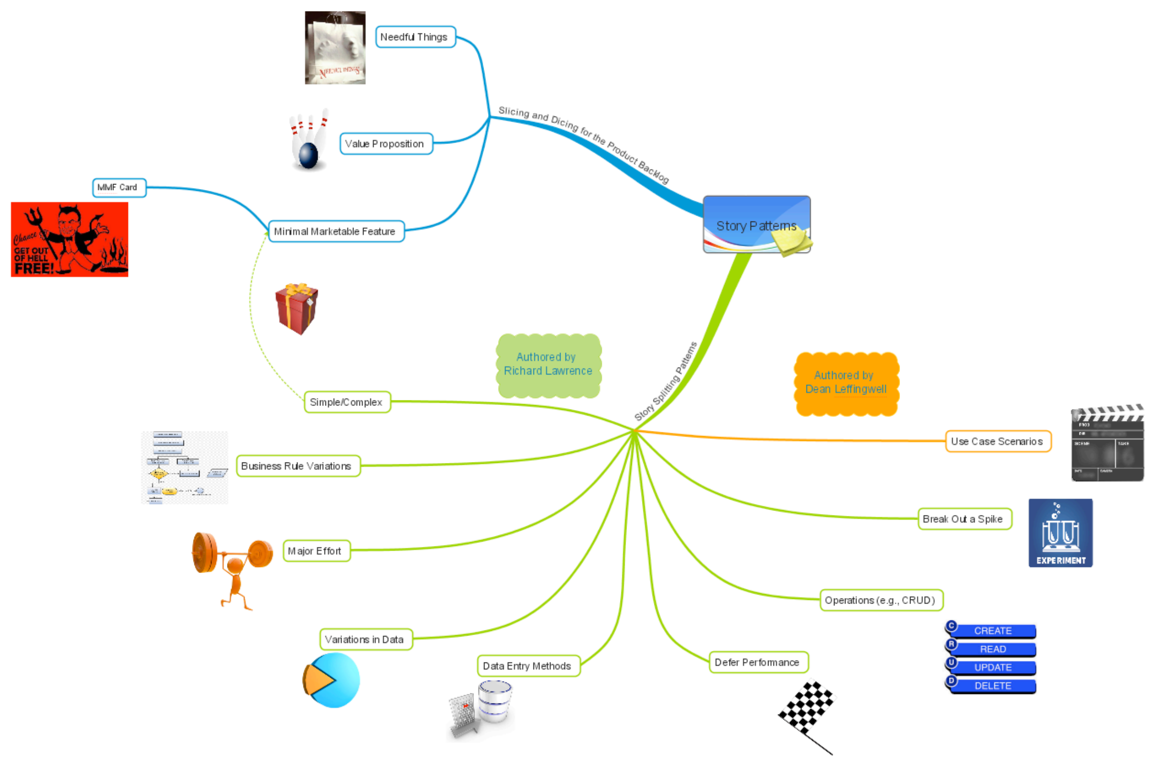


Figure 1. The Story Pattern Language Project MindMap

4. Patterns for the Workshop

Needful Things

Context

A long-established technique in Agile Development is to write stories to "represent requirements" (12). The purpose of a story is to drive (preferably, face-to-face) conversations to discover the requirements. Requirements can be functional, user, reliability, performance, serviceability or to do with a whole range of 'ilities' and this has led to a variety of story formats. However, confusion reigns when stories are also used to document requirements.

Forces

- It is well understood that product-project failures are caused by issues to do with requirements BUT asking users for system requirements is a 'solution' that does not work
- A product must be created which is fit-for-purpose BUT there are no system requirements to begin with
- Users are experts in their business operations BUT they are not typically experts in system requirements
- Developers have expertise in building products BUT are not typically experts in the domain that their products are used in

Problem

What is the scope of a story?

Solution

Write the story so that it expresses users' needs. Do not include system requirements that may (or may not) meet those requirements. System requirements are the outcome of the conversations driven by the story.

Resulting Context

The concerns of the users' needs, and the system requirements for meeting them, are clearly separated. The users' needs are captured in the story; the system requirements are captured separately in whatever documentation format (if any at all) the team determines is appropriate.

Rationale

The sole purpose of a story is to drive conversations between the Product Owner (or members of her Product team), between the developers and the customers, and between the members of the development team. The discussions clarify requirements, the business value of the story implemented in product, and the effort needed to develop it. These discussions should be allowed to develop freely, and this is best done by focussing on the users' needs at the start of the conversation rather than polluting with details of the system requirements to meet those needs.

Known Uses

The general advice in the literature concerning stories, for example Cohn (1), Leffingwell (10) and Rubin (13) has always been to write stories from the customers' perspective rather than from a technical viewpoint. Specifically, in 2013, a major global investment bank and, separately, a UK-based instrumentation company have both improved their requirements management processes by implementing a policy whereby stories are mandated to describe user needs rather than requirements.

Value Proposition

Context

Stories often begin as coarse-grained epics which are successively decomposed as they approach the build horizon. A typical issue that can occur is that the number of stories can grow so large that the team gets 'lost'. This is often symptomatic of having decomposed the stories too far.

Problem

How far should stories be decomposed?

Forces

- Stories often start as epics (compound stories) or as vague features which capture essential needs BUT are too large to be estimated or tested to any level of confidence
- Accurate effort estimation and testing require detailed requirements BUT the requisite detail is unavailable until epics and stories move closer to the build horizon
- A Product Backlog may contain "everything that might be needed in the product"(14) BUT is also an ordered list of things
- Ordering the Product Backlog items (PBIs) is a dynamic activity which lasts the full life cycle of the product as new learning emerges from the development cycle BUT the business value which should be the main driver for ordering can be obscured if there are too many stories to deal with at any one time

Solution

Break stories down only so far as their business value can be easily perceived by the customer. If their value is to the engineering effort rather than the customer, then they are likely candidates for the Sprint Backlog (to use Scrum's term for the current Sprint's development and construction plan) or a wider Engineering Backlog (if there is one) rather than the Product Backlog.

Resulting Context

The team is able to cleanly separate the concerns of the Product Backlog which contains value propositions recognizable by the customer, and any Backlog (such as the Sprint Backlog) which is primarily concerned with the engineering tasks that have to be performed to realize the value. The Product Backlog is now a list which is manageable for the purpose of ordering the PBIs.

Rationale

The Agile community often uses sloppy and misleading phraseology such as 'PBI's (or stories) need to be broken down into tasks'. Some tools such as Microsoft's TFS implement story/task hierarchies. However, if a hole in the ground is needed, in what sense is the digging of it a component of the hole? Tasks need to be performed to create business value, but they are not ‘valuable’ in that sense themselves. While tasks should be grouped according to the stories being developed, they should not themselves be represented in stories. Epics should rather be decomposed into discrete value propositions which are themselves finer-grained stories.

Known Uses

This pattern is strongly implied by the ‘V’ for ‘value’ in the INVEST acronym (3). The latest edition of The Scrum Guide (14) has added ‘value’ as one of the attributes needed in a minimum description of a PBI.

Minimal Marketable Feature

Context

A rule of thumb often referred to in patterns literature is to size PBIs so that they can be completed in a single Sprint. Frequently, however, teams are faced with epics which are not easily decomposed into Sprint-sized stories, typically because their technical complexities make it difficult to easily discern the 'seams' between potential stories

Problem

How can we start decomposing complex epics?

Forces

- An epic is simply a statement of a user need "in the large" BUT often this implies coarse-grained system requirements
- There is an overriding need to deliver 'early and often' so that the customer can extract value from the product BUT some value propositions are inherently complex
- Scrum requires that "potentially shippable product" be demonstrated at the end of every iteration BUT it may not make sense for some features to be delivered separately from others they depend on

Solution

Decompose epics first into Minimal Marketable Features (MMF). Continually ask (and answer) the question: "What is the smallest element of value that would be useful to the customer?" An MMF is defined as the smallest set of functionality that can be realized in order for the customer to perceive value. An MMF may itself be composed of more than one story, but typically most, if not all, of the stories which result from discussions about what constitutes an MMF are individually small enough to fit into a Sprint.

Resulting Context

Identification of the 'golden nugget(s)' of value in an epic allows for a more rational decomposition and subsequent ordering of the resulting stories. Product Owners can base their release planning on MMFs and give their constituent stories a higher priority in the build order than those features which add 'bells and whistles' to the product. Since Scrum requires product to be "potentially shippable" in the sense of its quality, but does not require it to be actually released until the Product Owner judges it is useful (to the customer) to do so, it allows both for the MMF to be incrementally built over many Sprints, and for further functionality to be layered onto them after initial release.

Rationale

The Agile Manifesto states that "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software". One of the biggest wastes in software development is the creation of features that are little or never used by the customer (On average 65% of delivered features according to one estimate (15)). The explicit identification of MMFs both focuses the team on what is truly valuable, mitigates against gold-plating, and provides a value-based criterion for the decomposition of epics into stories.

Known Uses

MMFs and/or similar ideas are in increasingly widespread use throughout the Agile community. There is a close relationship between MMF and 'Minimal Marketable Product' which has been an essential strategy in Apple's marketing of the iPhone and other innovative products (see Pilcher (16)) However, the concept of a Minimal Marketable Feature as a subset of a customer's requirements was developed first as a concept by Sun Microsystems and is detailed in the book Software By Numbers (17) in 2003. XP's James Shore has written about the use of MMF's for phased delivery in Agile product development (18). David Anderson uses the term Minimal Marketable Release (MMR) in Kanban (19)) and the Poppendiecks refer to Minimal Useful Feature Set (20) in their work on Lean Software Development.

MMF Card

Context

Stories typically start out as epics: rather coarse-grained, perhaps vague statements of user needs. They remain in the Product Backlog in that state until they are close enough to the build horizon for them to be decomposed. Often the epic is completely replaced in the Backlog and/or the card wall by the stories into which it has been decomposed, with the danger that the traceability between the original epic and these stories is now lost to the Team.

Problem

How can we maintain traceability between epics and stories?

Forces

- Epics are useful as placeholders before detailed requirements analysis is required BUT are insufficient for driving conversations closer to the build horizon
- Finer-grained stories drive detailed requirements work BUT may obscure the 'big picture' of the product
- Traceability between epics and stories is desirable BUT specialized traceability documentation is often wasteful in terms of the effort to create and maintain it.

Solution

Write MMF cards which list the component stories and maintain them alongside the story cards so that traceability is visualized. The format of MMF cards can vary. Denne et al (17) use a form in which the MMF is uniquely named and which states the value of the MMF. The individual stories are listed along with their effort estimations, and the sum of these estimations is ascribed to the MMF (see Fig 2 below for an example). Variations include the listing of all the stories related to the original epic, including those outside the boundary of the Minimum Marketable Feature, optionally including a reference to their priority also.

Resulting Context

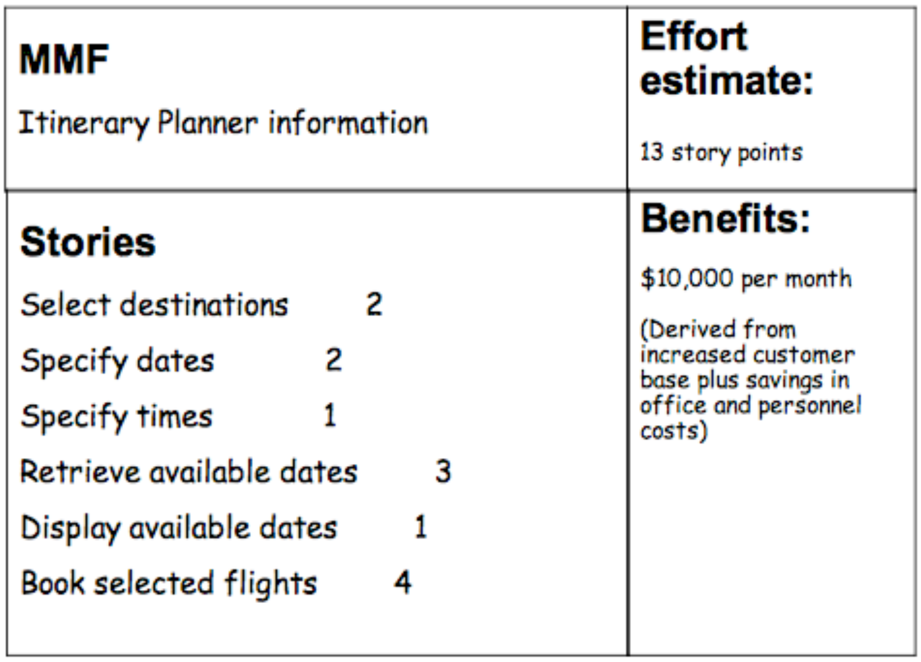
A top-down, hierarchical visualization of the relationship between epics and stories is maintained through the medium of MMF cards. This visualization facilitates the team's discussions as it iterates between the Sprint Planning horizon (where stories are most useful) and Release Planning when the coarser-grained epics and MMFs are typically better for driving conversations. The effort required to maintain traceability is minimized.

Rationale

Story cards are a proven and effective way of visualizing stories. They can be complemented by MMF Cards which summarize the Minimal Marketable Feature at a coarser level of granularity. In the life cycle of an epic the story card which represents it is decomposed into an MMF Card, plus optionally extra stories (i.e., those stories which are part of the epic but not its minimum value proposition). As the MMF is itself decomposed, new story cards appear alongside the MMF Card, but are also represented as a list on the MMF Card. (see Figure 2, below). Some teams use different colours to distinguish the different sizes of story being represented. Whatever the exact approach, the use of the MMF Card permits forward and backward traceability as teams drill down or, alternatively, abstract up through the different levels of granularity.

Known Uses

MMF Cards are an essential aspect of the Incremental Funding Model developed by Sun Microsystems. Emerald Hill Limited has introduced both the main variants described above to organizations in the hydrocarbons and financial investment industries. 21



MMF

Itinerary Planner information

Effort estimate:

13 story points

Stories

Select destinations	2
Specify dates	2
Specify times	1
Retrieve available dates	3
Display available dates	1
Book selected flights	4

Benefits:

$10,000 per month

(Derived from increased customer base plus savings in office and personnel costs)

Figure 2. An example MMF card as described in Denne et al (17)

References

(1) M. Cohn. 2004. User Stories Applied. Addison Wesley Professional

(2) R. Jeffries. 2001. "Essential XP: Card, Conversation, Confirmation" http://xprogramming.com/articles/expcardconversationconfirmation/

(3) W. C. Wake. 2003. "INVEST in Good Stories, and SMART Tasks". www.xp123.com

(4) J.Sutherland and K. Schwaber. The Scrum Guide. P16. February 2010

(5) According to the Seventh State of Agile Survey 2012 by VersionOne 74% of respondents were using Scrum or a Scrum hybrid

(6) Newer versions of The Scrum Guide were published in October 2011 and July 2013 respectively neither of which contained any reference to stories. This should not, however, be interpreted as Scrum 'dropping' or in any way deprecating the use of stories. Rather the Guide is trying to strip to a minimum the things which are mandatory for a team claiming to be applying the Scrum framework. Since PBI's need not be stories if a team decides not to use them, they are no longer included in The Scrum Guide.

(7) M. Beedle, M. Devos, Y. Sharon, K. Schwaber and J. Sutherland. 1996. "Scrum: A Pattern Language for Hyperproductive Software Development " Object Oriented Programming Languages and Systems convention. San Jose. California. USA

(8) K. Schwaber. 1995. "Scrum Development Process". In J. Sutherland (ed.) OOPSLA Business Object Design and Implementation Workshop. Springer

(9) R. Lawrence. 2009. "Patterns for Splitting Stories". www.richardlawrence.info/2009/10/28/patterns-for-splitting-user-stories

(10)D. Leffingwell. Agile Software Requirements. 2011. Pearson Education

(11) http://c2.com/cgi/wiki?PatternTemplate

(12) D. Lassig."User Stories, Epics, Themes and MVPs". blog.laessig.com.August 1, 2013

(13) K. Rubin.2013. Essential Scrum. Addison Wesley

(14) J. Sutherland and K. Schwaber. 2013. The Scrum Guide. P. 12. July 2013

(15) Standish Group. 1995-6. The CHAOS Report . The Standish Group

(16) R.Pilcher. 2013. "The Mimimal Viable Product and the Mimimum Marketable Product". www.romanpichler.com.October 9, 2013

(17) M. Denne and J. Cleland-Huang 2003. Software By Numbers: Low-risk, High-Return Development . Sun Microsystems

(18). J. Shore.2004. "Phased Releases" in The Art of Agile. www.jamesshore.com. January 1, 2004

(19) D,J, Anderson 2010. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Press

(20) M. Poppendieck and T. Poppendieck (2007) , Implementing Lean Software Development from Concept to Cash. Addison Wesley

(21) J. Patton. 2008. “The New User Story Backlog is a Map”. www.agileproductdesign.com. October 8, 2008

ACKNOWLEDGEMENTS

The author would like to record his special thanks for Anand Krishnaswamy’s insightful comments and helpful suggestions throughout the shepherding process for GuruPloP 2014. That the paper could even be considered for a Pattern Writers’ Workshop is in large part due to these.